PCT/IL 00 / 00744

09/720075

מדינת ישראל
STATE OF ISRAEL

IL00/744

משרד המשפטים
לשכת הפטנטים

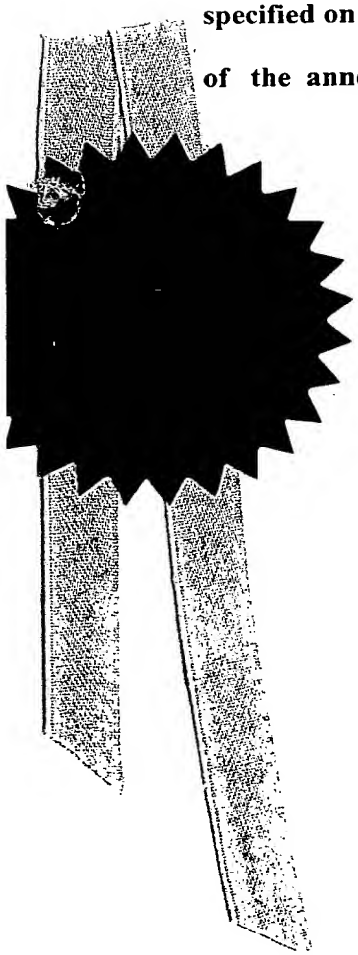
Ministry of Justice
Patent Office

4

זאת לתעודה כי
רצופים בזה העתקים
נכונים של המסמכים
שהופקדו לכתחילה
עם הבקשה לפטנט
לפי הפרטים הרשומים
בעמוד הראשון של
הנספח.

This is to certify that annexed hereto is a true copy of the documents as originally deposited with the patent application of which particulars are specified on the first page of the annex.

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)



This היום 19-11-2000

ממונה על הבוחנים

רשם הפטנטים
Commissioner of Patents

נתאשר
Certified

לשימוש הלשכה
For Office Use

מספר: 132929 Number	
תאריך: 14-11-1999 Date	
הוקדם/נדחה Ante/Post-dates	

חוק הפטנטים, התשכ"ז -- 1967
PATENTS LAW, 5727-1967

בקשה לפטנט
Application for Patent

C:35810

אני, (שם המבקש, מענו -- ולגבי גוף מאוגד -- מקום התאגדותו)
I (Name and address of applicant, and, in case of body corporate-place of incorporation)

YCD MULTIMEDIA
78 Bar-Kochva Street
Herzliya

ויי סי די מולטימדיה
רחוב בר-כוכבה 78
הרצליה

(An Israeli Company)

(חברה ישראלית)

Inventors: Noam Levavi & Danny Zeevi
(Israeli Citizens)

המציאים: נעם לבבי ודני זאבי
(אזרחים ישראלים)

בעל אמצאה מכח העברה שמה הוא By Assignment
Owner, by virtue of of an invention, the title of which is:

(בעברית) ממשק דינמי למשתמש
(Hebrew)

(באנגלית) DYNAMIC USER INTERFACE
(English)

מבקש בזאת כי ינתן לי עליה פטנט hereby apply for a patent to be granted to me in respect thereof

*בקשה חלוקה - Application for Division	*בקשת פטנט מוסף - Application for Patent of Addition	*דרישה דין קדימה Priority Claim		
מבקשת פטנט from Application	*לבקשה/לפטנט to Patent/Appl.	מספר/סימן Number/Mark	תאריך Date	מדינת האיגוד Convention Country
No._____ מס.	No._____ מס.			
dated_____ מיום	dated_____ מיום			
*יפוי כח: כללי/מיוחד - רצוף בזה / יוגש P.O.A.: general / individual - attached / to be filed later - filed in case_____ הוגש בענין				
המען למסירת הודעות ומסמכים בישראל Address for Service in Israel Sanford T. Colb & Co. P.O.B. 2273 Rehovot 76122				
חתימת המבקש Signature of Applicant For the Applicant, Sanford T. Colb & Co. C:35810		היום This 14 בחודש of November שנת of the year 1999		

לשימוש הלשכה
For Office Use

טופס זה, כשהוא מוטבע בחותם לשכת הפטנטים ומושלם בספר ובתאריך ההגשה, הינו אישור להגשת הבקשה שפרטיה רשומים לעיל.
This form, impressed with the Seal of the Patent Office and indicating the number and date of filing, certifies the filing of the application, the particulars of which are set out above.

* מחק את המיותר Delete whatever is inapplicable

ממשק דינמי למשתמש

DYNAMIC USER INTERFACE

ויי סי די מולטימדיה
הממציאים : נעם לבבי ודני זאבי
(אזרחים ישראלים)

YCD Multimedia
Inventors: Noam Levavi & Danny Zeevi
(Israeli Citizens)

C: 35810

Dynamic Skin - תאור הרעיון

בסיס הרעיון (הקונספט)

מתן תבנית ליצירת מסגרת כללית המאפשרת מימוש רחב של ממשק משתמש גרפי (GUI) ויכולה לשמש כל אפליקציה שהיא, ללא כל תלות במערכת ההפעלה עליה היא מורצת וללא כל שימוש בעורך (EDITOR) כלשהו המותאם לאותה האפליקציה, וזאת בהסתמך אך ורק על תמונות ומבנים גרפיים (צבעים, צורות, כתב וכו') שאותם ניתן ליצור ולערוך באמצעות כל תוכנת ציור בסיסית.

מטרת הסכמה (התבנית) המוצעת הנה מצד אחד לאפשר **שיתוף של אותם ממשקים גרפים בין אפליקציות שונות** (גם עבור אפליקציות ממערכות הפעלה שונות) ומצד שני לאפשר **לכל משתמש** (בעל ידע מינימלי של הפעלת תוכנת ציור בסיסית), לממש/לכתוב/להתאים/לשנות/לשתף כל ממשק גרפי בצורה מלאה, טבעית, פשוטה, קלה, חופשית, וגרפית לחלוטין.

נקודות מפתח ברעיון ה Dynamic Skin :

- המסגרת המוצעת אינה תלויה במערכת ההפעלה או באפליקציה מסוימת.
- המסגרת המוצעת אינה משתמשת (למימוש הממשק הגרפי) בכלי או עורך כלשהו.
- כלי המשתמש, למימוש הממשק הגרפי הנם גרפיים לחלוטין : תמונות ומבנים גרפיים (צבעים, צורות, כתב וכו') הניתנים למימוש או שינוי באמצעות כל תוכנת ציור בסיסית ועל כל מערכת הפעלה.
- המסגרת מגדירה "שפה גרפית" לתיאור **ממשק משתמש גמיש (Dynamic Skin)**, כך ששנויים אפשריים בממשק המשתמש יבוצעו ע"י שימוש בשפה זו – וזאת ע"י יצירה או עריכה של מידע גרפי בלבד.
- המסגרת משתמשת במידע גרפי סטנדרטי – תמונות דיגיטליות (דחוסות ולא דחוסות – ובכל תקן (תיאורי או גרפי), בכדי לתאר את ממשק המשתמש. שנויים בממשק הגרפי נעשים ע"י מניפולציה של התמונות הדיגיטליות בלבד (אין צורך ב Editor הספציפי לאפליקציה).
- המסגרת המוצעת מאפשרת הגדרה גמישה של **מספר בלתי מוגבל של אזורים גרפיים** (עד רמת הפיקסל הבודד) בממשק המשתמש, **המבצעים פעולה מסוימת אחת** באפליקציה ולחילופין **אזור גרפי יחיד** (עד רמת הפיקסל הבודד) יכול לבצע **מספר פעולות באפליקציה** (אם הוא עונה על מספר הגדרות גרפיות במקביל).
- המסגרת המוצעת באה לענות על הצורך של **שיתוף ממשקים גרפיים** בין מספר רב ככל האפשר של אפליקציות שונות – ללא צורך בבניית מתאמים מיוחדים למימוש פעולה זו.
- המסגרת המוצעת באה לענות על הצורך של שנוי והתאמת ממשק-משתמש (User Interface), **בצורה פשוטה מהירה וטבעית ככל האפשר, ע"י המשתמש בפועל (ה-USER)**.
- המסגרת **המוצעת מפרידה את שכבת מימוש ממשק המשתמש (GUI) מהאפליקציה.**
- המסגרת המוצעת מאפשרת גם למשתמש עצמו את **היכולת המלאה** של מימוש/שינוי/התאמת ממשק המשתמש.

הגדרת "השפה הגרפית", וסכמה כללית לאפליקציית Dynamic Skin:

המדיום המתאר את ממשק המשתמש הנו גרפי, משמע כל מידע הניתן להצגה במרחב דו-מימדי (ההרחבה עבור אוביקטים במרחב תלת-מימדי (3D)/ אובייקטים פנורמיים וכו' הנה טריביאלית – המצב נכון להיום הינו שהכלים לטיפול באוביקטים כאלו אינם כה פשוטים כמו תוכנת ציור במרחב דו-מימדי). המידע הגרפי מאוחסן בדרך כלל כקובץ תמונה על מחשב או בתוך כל כלי אחסון דיגיטלי אחר.

נגדיר מידע זה בתור ה"**מפה הגרפית**" (Graphic Map – GM) של ממשק המשתמש.

אחסון המידע הגרפי (הדיגיטלי) יכול להתבצע ע"י שימוש בקובץ תמונה לא דחוס (BITMAP) או ע"י קובץ תמונה דחוס (בפורמט כלשהו) או ע"י שימוש בכל פורמט אחר המתאר אובייקט של תמונה.

נגדיר "**אלמנט גרפי**" יחיד (GE – Graphic Element) בתור קבוצה חלקית כלשהי של נקודות מתוך **המפה הגרפית** או בתור קבוצה של קבוצות של נקודות אלו, וכן הלאה, כלומר (P = קבוצת החזקה של...):

[ראה איור 20]

אלמנט גרפי גם יכול להיות מוגדר כנקודה בודדת, כלומר:
(אלמנט גרפי שאינו קבוצה חלקית של המפה אלא שייך למפה כאיבר בודד)

[ראה איור 21]

<u>דוגמאות של הגדרת קבוצה גרפית:</u>

- אוסף הנקודות מתוך המפה הגרפית הנמצאות בתחום צבעים מסוים (צבע בודד, או רצף של צבעים) או קבוצה של אוספים כאלו (לדוגמא: אזור ירוק, או אזורים כהים).
- אוסף הנקודות הנמצאות בתוך, או כחלק מצורה גיאומטרית מסוימת, במפה הגרפית, או קבוצה של אוספים כאלו (לדוגמא: כוכב מחומש, או קבוצת כוכבים).
- אוסף הנקודות המגדירות קשר בין הצורה הגרפית שבה הן נמצאות לבין הצבע או טווח הצבעים שלה (לדוגמא: מתומו זהוב, או איסף מתומנים זהובים).
- אוסף הנקודות המקיף מילה מסויימת (המלה PLAY באנגלית לדוגמא).
- וכך הלאה...

Dynamic Skin – מבט על [ראה איור מס' 1]

הסכמה הכללית של אפליקציה בה משולבת טכנולוגית ה- Dynamic Skin:

1. אוסף של **אלמנטים גרפיים** מגדיר את ממשק המשתמש אותו ברצוננו ליצור, אלמנטים גרפיים אלו שוכנים בתוך **המפה הגרפית**.
2. **המפה הגרפית** מתקבלת ע"י טעינה של מדיום גרפי כלשהו (קבצי מחשב, או מדיום אחר) לזיכרון המחשב [ראה 1 באיור מס' 3].

2.1. תהליך הטעינה כולל המרה של המדיום הגרפי לקובץ Bitmap בזיכרון – קובץ זה הנו אוסף הנקודות המייצגות את **המפה הגרפית** בזיכרון המחשב [ראה 2 באיור מס' 3].

3. סריקה של המפה הגרפית וזיהוי אלמנטים גרפיים בתוכה [ראה 3 באיור מס' 3].

3.1. **זיהוי אלמנטים** הנו פונקציה או אלגוריתם כלשהו המבצע "הכרזה" על קבוצת או קבוצות נקודות בתור **אלמנט גרפי או אלמנטים גרפיים**, אלגוריתם זה נכתב ע"י בונה האפליקציה ויכול להיות שיוך נקודות ע"פ צבע או טווח צבעים, צורות גיאומטריות, סמלים, מלים, או כל התאמה אחרת.

3.2. אלגוריתם זיהוי האלמנטים הוא כזה הנותן חופש למשתמש להביע את רצונות העיצוב שלו (Design) בצורה חופשית ,גמישה ופשוטה ככל האפשר. על האלגוריתם לבצע את מלאכת התרגום וזיהוי האלמנטים הגרפיים אם ע"י זיהוי צבעים, צורות, כתב, יחסים בין סימבולים או בכל צורה אחרת.

3.3. כל **אלמנט גרפי** מתוך המפה ממופה לפונקציונליות-(מידע ואוסף פעולות) כלשהי בתוך האפליקציה. נגדיר פונקציונליות זו בתור "**אובייקט גרפי דינמי**" .

3.4. **האובייקט הגרפי הדינמי** מכיל את המידע הגרפי אותו מייצגות הנקודות, וכן את משמעותו של המידע, והדרכים בהן ניתן לעשות בו שימוש.

3.4.1. **אובייקט גרפי דינמי** הנו ייצוג בתוכנה של **אלמנט גרפי**.

3.4.2. האובייקט מגדיר את **מהותו** של אוסף הנקודות ,כיצד יבוא אוסף נקודות זה לכדי ביטוי על המסך , מהם הקשרים בין חברי הקבוצה, מהם הקשרים בין קבוצות אחרות ואילו אינטראקציות יבוצעו עם המשתמש. דוגמאות **לאובייקטים גרפיים דינמיים** יכולות להיות :

3.4.2.1. כפתור לחיץ (Push Button).
3.4.2.2. אזור גרירה (Slider).
3.4.2.3. אזור להצגת תמונה או רצף תמונות (אנימציה).
3.4.2.4. אזור להצגת טקסט.
3.4.2.5. Check Box.
3.4.2.6. כפתור רב מצבי.
3.4.2.7. אזור ציור חופשי Free Hand (מהווה גם קוצת אב ל Progress Bar)
3.4.2.8. כל מימוש אחר שהוא של GUI Control.

3.5. **האובייקטים הגרפיים הדינמיים** משמשים כמגשרים בין פעולות המשתמש ע"ג המסך לבין **האלמנטים הגרפיים** (שהנם לוגיים בלבד), הם האחראים על עדכון המסך והצגת המצבים השונים בהם הם עשויים להיות, למשתמש.

3.5.1. לדוגמא, **אובייקט גרפי דינמי** מסוג "כפתור לחיץ" אחראי על אחסון המידע הרלוונטי לכפתור (לחוץ \ לא לחוץ) הצגת הכפתור למשתמש במצבו הנוכחי, ותגובה לפעולות מצד המשתמש - עדכון מצב הכפתור כתגובה ללחיצה, ו"הודעה" לשאר האפליקציה שהכפתור נלחץ או שסמן העכבר עבר מעליו, למשל.

3.6. לאחר סריקת המפה הגרפית מתבצעת הגדרת הפונקציונליות של ממשק המשתמש.

3.6.1. הגדרה זו נעשית ע"י התאמה בין קבוצת **האלמנטים הגרפיים** לבין קבוצת **האובייקטים הגרפיים הדינמיים**, ההתאמה מתבצעת ע"י **פונקצית ההתאמה** [ראה 4 באיור מס' 3].

3.6.1.1. **פונקצית ההתאמה** הנה פונקציה הספציפית לכל אפליקציה. הפונקציה מקשרת **אובייקט גרפי דינמי** ל**אלמנט הגרפי** עליו הוא "אחראי" ומאתחלת את **האובייקט הגרפי הדינמי** במידע הדרוש לו לשם ביצוע הפעולות עליהן הוא מופקד, מידע זה מכיל את קבי הנקודות הרלוונטיות, ועיבודים שונים שלהן.

דוגמא להתאמה כזו – [ראה איור מס' 2]

4. הפעלת האפליקציה [ראה 5 באיור מס' 3].

4.1. במהלך פעולת האפליקציה מתרחשים אירועים גרפיים שונים, אירועים אלו כוללים :

- תזוזת עכבר ע"ג החלון.
- לחיצות עכבר ע"ג החלון.
- גרירה ע"ג החלון.
- צורך בעדכון מסך או כתיבת Text ע"ג החלון.
- כל אינטראקציה אחרת של המשתמש עם המחשב או אירוע עבורו אנו מעוניינים בשנוי גרפי כלשהו.

אירועים אלו מועברים מהאפליקציה ל**אובייקטים הגרפיים הדינמיים** הרלוונטיים, ואלו מעדכנים את המסך בהתאם, לדוגמא :

- אוסף נקודות כלשהו יתחלף באוסף נקודות אחר, למשל כפתור יילחץ בתגובה ללחיצה על המקש השמאלי בעכבר.
- תתבצע מניפולציה כלשהי ע"ג אוסף נקודות מסוים ,למשל ציור קו כלשהו או שנוי צבע הנקודות.

סכמה כללית מומלצת למיפוי תמונה לגרף בעזרת רעיון ה Dynamic Skin :

1. גרף הנו אוסף נקודות המקושרים ביניהם ע"י קשתות עם משקלים , הקשתות יכולות להיות כווניים או לא (תאור מלא ומדויק למהותם של גרפים, ניתן למצוא בתורת הגרפים). את הגרף ניתן לייצג ב - 2 צורות עיקריות : A. ע"י מטריצת קשרים B. ע"י רשימות קשרים.
2. אוסף הנקודות שלנו הנם הפיקסלים (Pixels) במרחב התמונה.
3. דרך ממומלצת לקשר נקודות בגרף היא ע"י קישור של נקודות בעלי צבע מוגדר או בעלי סט של צבעים מוגדרים והקרובות מרחק של פיקסל אחד אחת מהשניה (ישנם 8 פיקסלים שכנים לכל פיקסל נתון למעט הפיקסלים שבקצוות) באמצעות קשת לא כוונית עם משקל הנגזר מהצבע בסט.
4. בהנחה של מיפוי ע"פ קריטריון של סט הנקודות בתמונה שהנם בעלי אותו צבע, הגרף ימופה כך שיקושרו רק נקודות הסמוכות אחת לשניה מרחק של פיקסל בודד (בכל כוון כלשהו) ובעלות אותו צבע בדיוק. משקל הקשת בין הנקודות המקושרות בגרף יהיה 1. הגרף אינו כווני.

סכמה כללית מומלצת לקביעת ערך חד מימדי מתוך תמונה דו-מימדית (לדוגמא עבור מימוש של SLIDER פשוט) בעזרת רעיון ה Dynamic Skin :

1. יש למפות את התמונה לגרף.
2. ע"פ תורת הגרפים ניתן למצוא באם גרף ה SLIDER רציף(קשור) או לא.
3. ע"פ תורת הגרפים ניתן למצוא מסלולים כפולים.
4. בשלב ראשון יש לבדוק האם קיים על הגרף מסלול רציף בין נקודת ההתחלה לנקודת הסוף.
5. במידה ולא קיים מסלול רציף – ימופה מסלול וירטואלי של קו ישר בין 2 הנקודות (נקודת ההתחלה ונקודת הסוף) להלן יקרא מסלול זה כ **"המסלול הליניארי הוירטואלי"** . ערך כל נקודה על הגרף יהיה לפי הערך של הנקודה הקרובה ביותר אליו ושנמצאת על **"המסלול הליניארי הוירטואלי"**.
6. ערך הנקודה על **"המסלול הליניארי הוירטואלי"** הנו שבר שמבטא את המרחק של הנקודה מנקודת ההתחלה יחסית לאורך המסלול – זהו מקרה ליניארי (קו ישר)
7. במידה וקיים מסלול רציף ייקבע מסלול ייחוס שהנו הדרך הקצרה ביותר בין ההתחלה לבין הסוף לפי אלגוריתם "SHORTEST PATH" מתורת הגרפים להלן יקרא מסלול זה **"המסלול הקצר ביותר"**. מסלול זה ניתן להיקבע חד ערכית. ערך כל נקודה על הגרף יהיה לפי הערך של הנקודה הקרובה ביותר אליו ושנמצאת על **"המסלול הקצר ביותר"**.
8. ערך הנקודה על **"המסלול הקצר ביותר"** הנו שבר שמבטא את המרחק של הנקודה מנקודת ההתחלה לאורך אותו מסלול נבחר וזאת יחסית לאורך המסלול הקצר ביותר כולו.
9. כל נקודה בגרף מוצאת את הנקודה המתאימה לה אם על ה **"המסלול הליניארי הוירטואלי"** ואם על ה **"המסלול הקצר ביותר"** ע"י שימוש באלגוריתם שמדמה שליחת מספר רב של קרניים (Ray Tracing) במרחב התמונה (בכל הכוונים עם הפרש קבוע ביניהם – כמות מומלצת של 16 קרניים יחלקו את המרחב לפרוסות בזוויות של 22.5 מעלות בין כל קרו) ומציאת הנקודה הקרובה ביותר שאחת הקרניים פוגשת ושנמצאת על אחד המסלולים. ניתן לממש מנגנון זה גם בדרכים אחרות.
10. מנגנון זה מאפשר גם התאמות של ערך חד מימדי עבור תמונות שונות.
11. שחזור מיקום ע"פ ערך חד מימדי עבור מקרה של **"המסלול הקצר ביותר"** הנו טריביאלי ונעשה באופן הבא : לכל נקודה על המסלול הקצר יש ערך יחסי למרחקו מתחילת המסלול – השיוך נעשה לנקודה עם הערך הקרוב ביותר לערך החד מימדי.
12. שחזור מיקום ע"פ ערך חד מימדי עבור מקרה של **"המסלול הליניארי הוירטואלי"** הנו יותר מורכב ונעשה ע"י כך שלכל נקודה בגרף מקבלת את הערך היחסי של הנקודה הקרובה ביותר אליה הנמצאת על **"המסלול הליניארי הוירטואלי"** – השיוך נעשה לנקודה (על הגרף) עם הערך הקרוב ביותר לערך החד מימדי.

סכמה כללית מומלצת לקביעת ערך דו-מימדי מתוך תמונה דו-מימדית ולדוגמא עבור מימוש של אובייקט הנע לאורך מספר מסלולים - SUPER SLIDER פשוט בעזרת רעיון ה Dynamic Skin :

1. הערך ישמר בתור שני שברים יחסיים למיקום X ו Y על פני התמונה.
2. שחזור מיקום ע"פ ערך דו מימדי עבור נעשה כך שנבחרת נקודה על פני המסלול הקרובה ביותר לנקודה שנשמרה (היחסית) .
3. דרך מומלצת למציאת הנקודה הקרובה : ע"י שליחת קרניים בכל הכוונים במרחב התמונה (כמות קרניים מומלצת: 16) ומציאת הנקודה הקרובה ביותר שאחת הקרניים פוגשת ושנמצאת על אחד המסלולים. ניתן לממש מנגנון זה גם בצורות אחרות.

סכמות כלליות ליצירת אלמנטים גרפיים (Window Controls) גמישים בעזרת רעיון **Dynamic Skin** :

הדוגמאות הבאות מתארות מימושים אפשריים לאלמנטים גרפיים, המימושים מתוארים באופן כללי כדי לא "לכבול" אותם לאפליקציה ספציפית או מערכת הפעלה ספציפית. המימושים נכתבו בהנחה שלמערכת ההפעלה יכולת גרפית "חלונאית".

מערכות הפעלה נפוצות התומכות ביכולות הנדרשות ע"י מימושים אלו הן :

- Windows 3.11
- Windows 95/98
- Windows NT
- Sun Solaris
- OpenVMS
- Linux
- "Java Virtual Machine" running on any Java Enabled Machine

התאמה לסכמה הכללית :

כל אחד מן המימושים עוקב באופן כללי אחרי ה "הסכמה הכללית של אפליקציה בה משולב "**Dynamic Skin**" שתוארה בתחילת המסמך.
כל המימושים של רעיון ה"**Dynamic Skin**" פועלים "פר חלון", עבור חלונות אפליקציה גרפיים, כלומר לא כל האפליקציה חייבת לממש אלמנטים חלונאים בעזרת הרעיון.

אפשרות לאלמנטים גרפיים מסוגים שונים עבור כל חלון :

כל מימוש המתואר להלן הנו של **אלמנט גרפי בודד**, וזאת על מנת לפשט את התיאור של כל אחד מהם. באפליקציה "אמיתית" המצב הנפוץ הוא, שישנם מספר אלמנטים גרפיים הנמצאים במקביל בתוך האפליקציה – מצב זה נתמך, כמובן, ע"י המימושים המתוארים.

תאור מנגנון שכבות אפשרי :

עבור האלמנטים הגרפיים ייתכן מנגנון שכבות המציין "היררכית ציור" ביניהם. לדוגמא הרקע השקוף יצויר ראשון, לאחריו הכפתורים ולאחר מכן ה Slider –ים, מנגנון כזה ימנע מצב בו האלמנטים "עולים" זה על זה באופן אקראי. מנגנון זה הנו חלק מן האפליקציה המשתמשת ברעיון ה "**Dynamic Skin**" וממש היררכיה ספציפית.

קובץ "מפה גרפית" רצוי :

סוג הקובץ הרצוי עבור ה"מפה הגרפית" הנו קובץ 24Bit Bitmap (RGB) .

תמונות הדוגמא :

תמונות הדוגמא נלקחו מאפליקצית נגן מוזיקה ממוחשב.

סכמה כללית למימוש "חלון שקוף" גמיש בעזרת רעיון Dynamic Skin

1. יצירת / שימוש בקובץ תמונה מלבני (אין פורמט אחר לקובץ) המייצג את הצורה והרקע (Background) של חלון האפליקציה:
   - הקובץ יכיל את המידע הגרפי (צורות וצבעים) אותו אנו מעוניינים להראות ברקע של חלון האפליקציה.
   - הקובץ יהווה **מפה גרפית** בה יהיה **אלמנט גרפי** אחד שייצג את החלון, **אלמנט גרפי** זה יוגדר בתור:
     - "קבוצת כל הנקודות שאינן צבועות בטווח צבעים מסוים", במידה וזהו האלמנט הגרפי היחיד באפליקציה.
     - "קבוצת כל הנקודות שאינן צבועות בטווח צבעים מסוים ואינו מהווה **אלמנט גרפי** אחר" במידה וקיימים אלמנטים גרפיים נוספים באפליקציה אותם אנו רוצים להציג בחלון.

   אלמנט גרפי זה יוגדר בתור **אלמנט הרקע**.
   טווח הצבעים יוגדר ע"י כותב האפליקציה (למשל סגול בהיר).

2. המרת הקובץ לפורמט Bitmap וטעינתו לזיכרון המחשב.

3. סריקת הקובץ לשם מציאת הנקודות המהוות את **אלמנט הרקע**.

3.1. לאחר הסריקה **תוצמד** קבוצת הנקודות של **אלמנט הרקע** לאובייקט "רקע" באפליקציה, אובייקט זה יהיה אחראי על ציור הרקע של החלון בכל פעם שיידרש לכך (עדכוני מסך, גרירת חלון וכו').

3.2. עם יצירתו יאתחל אובייקט **אלמנט הרקע** ב"אוסף הנקודות" הרלוונטי אליו, כמו כן "אזור החלון" של האפליקציה (אזור החלון הנראה לעין המשתמש) יוגדר (ע"י קריאות API) כך שכלול רק את הנקודות הנמצאות **באלמנט הרקע**. כך יוצר מצב בו חלון האפליקציה יקבל את **צורת** אלמנט הרקע ושאר השטח המלבני יהיה למעשה, שקוף.

4. הפעלת האפליקציה:

4.1. לאחר טעינת האפליקציה מתבצעות אחת או יותר מן הפעולות הבאות (וזאת בהתאם לפעולות המשתמש ולמימוש הספציפי של האפליקציה), וכן כל פעולה אחרת המערבת את הנקודות הרלוונטיות ושאותה יכול להגדיר כותב האפליקציה:
   - כתגובה לבקשת עדכון מסך: העתקת קבוצת הנקודות המתאימה **לאלמנט הרקע** מהקובץ המדובר אל חלון האפליקציה.
   - כתגובה לגרירת החלון ע"י המשתמש: העתקת קבוצת הנקודות המתאימה **לאלמנט הרקע** מהקובץ המדובר אל חלון האפליקציה, זאת עבור כל תזוזה של החלון.

<u>דוגמא:</u>

כאן **אלמנט הרקע** הוא קבוצת כל הנקודות
שאינן צבועות בצבע סגול (במידה ולא הוגדרו אלמנטים נוספים):
[ראה איור מס' 4]

חלון האפליקציה כפי שנראה בעת פעולה:
[ראה איור מס' 5]

סכמה כללית למימוש "כפתורים רב מצביים" (Multi Buttons) גמישים בעזרת רעיון Dynamic Skin

1. יצירת / שימוש ב שני קבצי תמונה :

- קובץ "**רצועת מצבים**" (Stripe) – הקובץ מתאר (בציור \ תמונה) את מגוון המצבים הויזואליים בהם יכול להיות נתון הכפתור, המצבים שוכנים זה לצד זה ברצועה.
כל מצב הוא מלבן בגודל קבוע המכיל את תמונת הכפתור.
- כל אחד מן המצבים כולל בתוכו **אלמנט רקע** (ראה סעיף "חלון שקוף") אותו אנו רוצים לכלול בהצגת הכפתור.
- קובץ **המפה הגרפית** (נבחר בשם קובץ של **DSkin** : DSkin = Dynamic Skin) המגדיר את האלמנטים הגרפיים, במקרה הזה "**כפתורים רב מצביים**".
- כפתור רב מצבי, בסכמה זו, מוגדר כקבוצה של קבוצות של נקודות בעלות בצבעים מסוימים, קבוצה זו מוגדרת בצורה הבאה :
  - לכל **כפתור רב מצבי** מותאם טווח צבעים, טווח זה הנו רציף.
  - כל קבוצת נקודות הצבועות באחד מצבעי הטווח, מלבד הצבע האחרון, תוגדר כ "**אזור החלטה**" נפרד.
  - הצבע האחרון בטווח יוגדר כ "**צבע בסיס**"
  - קבוצת הנקודות המגדירה את המלבן המינימלי המכיל נקודות **בצבע הבסיס** תוגדר כ "**אזור צביעה**".

לכל **כפתור רב מצבי** מוגדר טווח צבעים ספציפי, והנקודות הרלוונטיות לכפתור זה צבועות בצבע זה.
המצבים השונים של כל כפתור רב מצבי יילקחו מרצועת המצבים ויצוירו **בתוך אזור הצביעה**.

[הערה : כאן מתוארת התאמה מדוייקת בין הנקודות באזור הצביעה, והנקודות ברצועת המצבים אולם תיתכן גם התמרה, בין הנקודות המהוות מצב בודד מרצועת המצבים לבין **אזור הצביעה** בתמונת ה - DSkin . המשמעות היא שאין תלות בין אזורי ההחלטה לאיזורי הצביעה].

רצוי שקובץ ה DSkin יהיה מאוחסן כפורמט גרפי שאינו גורם לשנוי המידע בתוכו (כדוגמת קובץ bitmap - BMP . לדוגמא : הקובץ יכול להיות DSkin.bmp) וזאת בכדי לשמור על החד-ערכיות של אזורי הצבע.

2. המרת הקבצים לפורמט Bitmap וטעינת שני הקבצים לזיכרון המחשב.

3. סריקת קובץ ה DSkin וחיפוש אזורי צבע ספציפיים בתוכו :
3.1. האפליקציה תחפש אזורים בעלי קודים מסוימים של צבעים הרלוונטיים לאפליקציה.
3.2. לאחר הסריקה תוצמד כל קבוצת של קבוצות נקודות בתחום הצבעים הרלוונטי לאובייקט "**כפתור רב מצבי**" באפליקציה, אובייקט זה יהיה אחראי על ציור הכפתור בכל פעם שיידרש לכך (עדכוני מסך, לחיצות משתמש וכו').
3.3. אובייקט "**כפתור רב מצבי**" ישויך לכל אחד מטווחי הצבעים הרלוונטיים ויממש את הפונקציונליות הנדרשת ממנו באפליקציה (למשל שנוי ערך כלשהו\ וכו').
3.4. עם יצירתו יאתחל כל אובייקט "**כפתור רב מצבי**" ב"**אוסף הנקודות**" הרלוונטי אליו.
3.4.1. כל "**כפתור רב מצבי**" ממפה את טווח הצבעים שלו ל**רצועת המצבים** שלו , כך שלכל צבע בטווח מתאים מצב מ**רצועת המצבים**.
3.5. כמו כן תחזיק האפליקציה מבנה נתונים שימפה בין טווח הצבעים של כל **אלמנט גרפי** לבין אובייקט ה"**כפתור רב מצבי**" הרלוונטי לטווח צבעים זה.

<u>דוגמא :</u>

קובץ **רצועת מצבים**
[ראה איור מס' 6]

קובץ **מפה גרפית** עבור כפתור רב מצבי
[ראה איור מס' 7]

4. הפעלת האפליקציה :

4.1. לאחר טעינת האפליקציה מתבצעות אחת או יותר מן הפעולות הבאות (וזאת בהתאם לפעולות המשתמש ולמימוש הספציפי של האפליקציה, וכן כל פעולה אחרת המערבת את הנקודות הרלוונטיות ושאותה יכול להגדיר כותב האפליקציה) :

- תגובה לבקשת עדכון מסך : ציור ה"**כפתור הרב מצבי**" באזור הצביעה (כפי שהוגדר והושם בסריקה), ציור זה נעשה ע"י העתקת נקודות אל **אזור הצביעה** בכל "**כפתור רב מצבי**" שהוגדר.
  העתקה זו נעשית ממלבן המצב המתאים בתמונת **רצועת המצבים** אל חלון האפליקציה. בחירת המצב המתאים מתוך הרצועה נעשית לפי מצב הכפתור.
  מצב הכפתור, כאמור, מאוחסן באובייקט ה"**כפתור הרב מצבי**" הספציפי.
- כתגובה ללחיצת המשתמש על **אזור החלטה** :
  - שנוי המצב של הכפתור הרב מצבי כך שיתאים **לאזור ההחלטה** עליו לחץ המשתמש.
  - העתקת הנקודות הרלוונטיות למצב זה מ**רצועת המצבים** אל **אזור הצביעה**.

סכמה כללית למימוש "אלמנטי גרירה" (Slider Controls) גמישים בעזרת רעיון Dynamic Skin

1. יצירת / שימוש ב - שני קבצי תמונה מלבניים:
   - הקובץ הראשון יהווה **מפה גרפית** (DSkin) בה יהיו אחד או יותר **אלמנטים גרפיים**. אלמנטים גרפיים אלו יכילו את קבוצת כל הנקודות הצבועות בטווח צבעים מסוים (לדוגמא – כל הנקודות הכחולות) כל אחד מאלמנטים גרפיים אלו יוגדר בתור **אלמנט גרירה**.
     טווח הצבעים עבור כל אלמנט גרירה יוגדר ע"י כותב האפליקציה.
   - הקובץ השני יוגדר עבור כל אחד **מאלמנטי הגרירה**.
   - קובץ זה יכיל **אלמנט רקע בחד** שיהווה את הצורה הנגררת, אלמנט רקע זה יוגדר בתור **סמן הגרירה**.

   רצוי שקובץ ה DSkin יהיה מאוחסן כפורמט גרפי שהנו (כדוגמת קובץ - bitmap BMP
   אשר אינו גורם לשנוי המידע בתוכו) וזאת בכדי לשמור על החד-ערכיות של אזורי הצבע.
2. המרת הקבצים לפורמט Bitmap וטעינת 2 הקבצים לזיכרון המחשב.
3. סריקת קובץ ה DSkin וחיפוש אזורי צבע ספציפיים בתוכו:
   3.1. האפליקציה תחפש אזורים בעלי קודים מסוימים של צבע הרלוונטיים לאפליקציה.
   3.2. לאחר הסריקה תוצמד כל קבוצה של נקודות השייכות לאותו **אלמנט גרפי** לאובייקט "**Slider**" באפליקציה, אובייקט זה יהיה אחראי על-ציור סמן הגרירה בכל פעם שיידרש לכך (עדכוני מסך, לחיצות משתמש, גרירת הסמן וכו').
   3.3. אובייקט "**Slider**" ישויך לכל אחד מטווחי הצבעים הרלוונטיים ויממש את הפונקציונליות הנדרשת ממנו באפליקציה (למשל שנוי ערך כלשהו וכו').
   3.4. עם יצירתו יאתחל כל אובייקט "**Slider**" ב"**אוסף הנקודות**" הרלוונטי אליו:
      3.4.1. כל "**Slider**" יאתחל **בסמן הגרירה** הרלוונטי אליו.
      3.4.2. כל "**Slider**" ימפה את הנקודות הרלוונטיות שלו לתוך מבנה נתונים המייצג גרף, ויצור **מסלול** עבור גרף זה. (ראה "סכמה כללית מומלצת למיפוי תמונה לגרף").
      3.4.3. האובייקט יכיל אלגוריתמים לקביעת מיקום הסמן ע"פ ערך מספרי או אחר המוצמד ע"י האפליקציה לאובייקט ה "**Slider**" (ראה "סכמה כללית מומלצת לקביעת ערך חד מימדי מתוך תמונה דו-מימדית" וכן "סכמה כללית מומלצת לקביעת ערך דו-מימדי מתוך תמונה דו-מימדית").
   3.5. כמו כן תחזיק האפליקציה מבנה נתונים שימפה בין טווח הצבעים של כל **אלמנט גרפי** לבין אובייקט ה"**Slider**" הרלוונטי לטווח צבעים זה.
4. הפעלת האפליקציה:
   4.1. לאחר טעינת האפליקציה מתבצעות אחת או יותר מן הפעולות הבאות (וזאת בהתאם לפעולות המשתמש ולמימוש הספציפי של האפליקציה), וכן כל פעולה אחרת המערבת את הנקודות הרלוונטיות ושאותה יכול להגדיר כותב האפליקציה:
      - כתגובה לבקשת עדכון מסך:
        - העתקת קבוצת הנקודות המתאימה לסמן הגרפי (כפי שהוגדרה והושמה לתוך אובייקט ה "**Slider**" בסריקה) מתמונת הסמן אל מיקומו הנוכחי, כפי שנרבע ע"י אובייקט ה "**Slider**", בחלון האפליקציה, וזאת לפי מצב ה "**Slider**". המצב הנוכחי מאוחסן באובייקט ה"**Slider**" הספציפי.
      - כתגובה ללחיצת המשתמש על נקודה **במסלול** ה "**Slider**": העתקת קבוצת הנקודות המתאימה לסמן הגרפי (כפי שהוגדרה והושמה לתוך אובייקט ה "**Slider**" בסריקה) מתמונת הסמן אל מיקומו הנוכחי (כפי שחושב ע"י האלגוריתם) **במסלול**.
      - כתגובה לגרירה של סמן ה "**Slider**" ע"י המשתמש: העתקת קבוצת הנקודות המתאימה לסמן הגרפי (כפי שהוגדרה והושמה לתוך אובייקט ה "**Slider**" בסריקה) מתמונת הסמן אל מיקומו הנוכחי (כפי שחושב ע"י האלגוריתם) במסלול. ההעתקה היא כזו שתגרום ל"גרירה רציפה" של הסמן לאורך **המסלול** שלו (ראה "סכמה כללית מומלצת לקביעת ערך חד מימדי מתוך תמונה דו-מימדית" וכן "סכמה כללית מומלצת לקביעת ערך דו-מימדי מתוך תמונה דו-מימדית").

דוגמא :

קובץ DSkin (ראה סעיף 1) **עם אלמנט גרירה**
[ראה איור מס' 8]

קובץ ובתוכו **סמן הגרירה** (סמן הגרירה מוגדל יחסית לגודלו היחסי בתמונה)
[ראה איור מס' 9]

קובץ המראה את חלון האפליקציה ובתוכו סמן הגרירה
[ראה איור מס' 10]

סכמה כללית למימוש "אזור צביעה חופשי" (Free Hand Painting Area) גמיש בעזרת רעיון Dynamic Skin

1. שימוש בשלושה קבצים גרפיים :
   - הקובץ הראשון יהווה **מפה גרפית** (DSkin) בה יהיו אחד או יותר **אלמנטים גרפיים**. אלמנטים גרפיים אלו יכילו את קבוצת כל הנקודות הצבועות בטווח צבעים מסוים (לדוגמא – כל הנקודות הכחולות) כל אחד מאלמנטים גרפיים אלו יוגדר בתור **אזור צביעה חופשית**. לחלופין, הגדרת **אזור צביעה חופשית** יכולה להיות : "אוסף הנקודות ברבוע המינימלי המכיל טווח צבעים מסוים", לדוגמא – אוסף הנקודות השייכות לרבוע המינימלי המכיל נקודות בצבע כחול.
   - שני הקבצים האחרים יהיו זהים בגודלם ויהוו את הרקע (Background color pattern) ואת ה"צבע הקדמי" (Foreground Color Pattern) של **אזור הצביעה החופשית**.
2. המרת הקבצים לפורמט Bitmap וטעינת שני הקבצים לזיכרון המחשב.
3. סריקת קובץ ה DSkin וחיפוש אזורי צבע ספציפיים בתוכו :
   3.1. האפליקציה תחפש אזורים בעלי קודים מסוימים של צבע הרלוונטיים לאפליקציה.
   3.2. לאחר הסריקה תוצמד כל קבוצה של נקודות השייכות לאותו **אלמנט גרפי** לאובייקט "**אזור צביעה חופשית**" באפליקציה, אובייקט זה יהיה אחראי על קבוצת הנקודות הכלולות באלמנט הגרפי ועל הציור העדכני בתוך **אזור הצביעה החופשית** (עדכוני מסך, לחיצות משתמש באזור, גרירת העכבר וכו').
   3.3. אובייקט " **אזור צביעה חופשית** " ישויך לכל אחד מטווחי הצבעים הרלוונטיים ויממש את הפונקציונליות הנדרשת ממנו באפליקציה (למשל ציור קו בין כל הנקודות באלמנט הרלוונטי בהן עובר העכבר).
   3.4. עם יצירתו יאתחל כל אובייקט " **אזור צביעה חופשית** " ב"**אוסף הנקודות**" הרלוונטי אליו :
      3.4.1. כל " **אזור צביעה חופשית** " יאתחל בקובץ הרקע ובקובץ הצבע הקדמי הרלוונטיים אליו.
      3.4.2. כל " **אזור צביעה חופשית** " ימפה את הנקודות הרלוונטיות שלו לתוך מבנה נתונים המייצג את מרחב הנקודות, למשל מערך של נקודות, כך שתתאפשר שמירה של הנקודות אותן אנו רוצים לצבוע בצבע הרקע או בצבע הקדמי.
   3.5. כמו כן תחזיק האפליקציה מבנה נתונים שימפה בין טווח הצבעים של כל **אלמנט גרפי** לבין אובייקט ה" **אזור צביעה חופשית** " הרלוונטי לטווח צבעים זה.
4. הפעלת האפליקציה :
   4.1. לאחר טעינת האפליקציה מתבצעות אחת או יותר מן הפעולות הבאות (וזאת בהתאם לפעולות המשתמש ולמימוש הספציפי של האפליקציה), וכן כל פעולה אחרת המערבת את הנקודות הרלוונטיות ושאותה יכול להגדיר כותב האפליקציה :
      - כתגובה לבקשת עדכון מסך :
        - העתקת קבוצת הנקודות עבור רקע אזור הצביעה (כפי שהוגדרה והושמה לתוך אובייקט ה "**אזור צביעה חופשית** " בסריקה) מתמונת הרקע אל **אזור הצביעה החופשית** (תיתכן התמרה של הנקודות במידה והגדלים אינם מתאימים) בחלון האפליקציה, וזאת לפי מרחב הנקודות המיוצג באובייקט " **אזור הצביעה החופשית** " הרלוונטי.
        - העתקת קבוצת הנקודות עבור "קדמת" אזור הצביעה (כפי שהוגדרה והושמה לתוך אובייקט ה **אזור צביעה חופשית**" " בסריקה) מתמונת ה"צבע הקדמי" אל **אזור הצביעה החופשית** (תיתכן התמרה של הנקודות במידה והגדלים אינם מתאימים) בחלון האפליקציה, וזאת לפי מרחב הנקודות המיוצג באובייקט " **אזור הצביעה החופשית** " הרלוונטי.
      - כתגובה ללחיצת המשתמש על נקודה **בתוך אזור הצביעה החופשית** : העתקה של אחת או יותר נקודות מאחת מתמונות הרקע או הצבע הקדמי אל חלון האפליקציה, וזאת לפי מבנה הנתונים המייצג את מרחב הנקודות ואו אלגוריתם ציור כזה או אחר.
      - כתגובה לגרירת העכבר על פני המשטח: העתקה של אחת או יותר נקודות מאחת מתמונות הרקע או הצבע הקדמי אל חלון האפליקציה, וזאת לפי מבנה הנתונים המייצג את מרחב הנקודות ואו אלגוריתם ציור כזה או אחר, העתקה זו צריכה להתבצע באופן שיראה "ציור רציף" של גרפיקה.

דוגמא :

מפה גרפית ובתוכה אלמנט מסוג "אזור צביעה חופשית" (במקרה זה)

[ראה איור 11]

קובץ רקע לאלמנט "אזור צביעה חופשית"

[ראה איור 12]

קובץ "צבע קדמי" לאלמנט "אזור צביעה חופשית"

[ראה איור 13]

חלון האפליקציה עם אלמנט "אזור צביעה חופשית" בתוכו

[ראה איור 14]

סכמה כללית למימוש "אזורי הצגת טקסט" (Static Text Controls) גמישים בעזרת רעיון Dynamic Skin

1. שימוש בקובץ גרפי בודד:
   - הקובץ יהווה **מפה גרפית** (DSkin) בה יהיו אחד או יותר **אלמנטים גרפיים**. אלמנטים גרפיים אלו יכילו את קבוצת כל הנקודות הצבועות בטווח צבעים מסוים (לדוגמא – כל הנקודות הכחולות) כל אחד מאלמנטים גרפיים אלו יוגדר בתור **אזור הצגת טקסט**. לחלופין, הגדרת **אזור הצגת טקסט** יכולה להיות: "אוסף הנקודות ברבוע המינימלי המכיל טווח צבעים מסוים", לדוגמא – אוסף הנקודות השייכות לרבוע המינימלי המכיל נקודות בצבע אדום.
2. המרת הקבצים לפורמט Bitmap וטעינת שני הקבצים לזיכרון המחשב.
3. סריקת קובץ ה DSkin וחיפוש אזורי צבע ספציפיים בתוכו:
   3.1. האפליקציה תחפש אזורים בעלי קודים מסוימים של צבע הרלוונטיים לאפליקציה.
   3.2. לאחר הסריקה תוצמד כל קבוצה של נקודות השייכות לאותו **אלמנט גרפי** לאובייקט "**אזור הצגת טקסט**" באפליקציה, אובייקט זה יהיה אחראי על קבוצת הנקודות הכלולות באלמנט הגרפי ועל הצגת הטקסט בתוך **אזור הצגת הטקסט** (עדכוני מסך, שנוי ערך הטקסט, גרירת העכבר וכו').
   3.3. אובייקט "**אזור הצגת טקסט**" ישויך לכל אחד מטווחי הצבעים הרלוונטיים ויממש את הפונקציונליות הנדרשת ממנו באפליקציה , למשל: הצגת טקסט רלוונטי ע"ג הנקודות הכלולות באובייקט, וזאת ע"י קריאות API של מערכת ההפעלה.
   3.4. עם יצירתו יאתחל כל אובייקט "**אזור הצגת טקסט**" ב"**אוסף הנקודות**" הרלוונטי אליו.
   3.5. כל אובייקט "**אזור הצגת טקסט**" יחזיק מבנה נתונים שיכיל את הטקסט אותו הוא יראה ע"ג חלון האפליקציה.
   3.6. כמו כן תחזיק האפליקציה מבנה נתונים שימפה בין טווח הצבעים של כל **אלמנט גרפי** לבין אובייקט ה"**אזור הצגת טקסט**" הרלוונטי לטווח צבעים זה.
5. הפעלת האפליקציה:
   5.1. לאחר טעינת האפליקציה מתבצעות אחת או יותר מן הפעולות הבאות (וזאת בהתאם לפעולות המשתמש ולמימוש הספציפי של האפליקציה), וכן כל פעולה אחרת המערבת את הנקודות הרלוונטיות ושאותה יכול להגדיר כותב האפליקציה:
      - כתגובה לבקשת עדכון מסך:
        - ציור הטקסט הרלוונטי ל אובייקט "**אזור הצגת טקסט**" בתוך תחום הנקודות הרלוונטי אליו, ע"ג חלון האפליקציה. ציור הטקסט יתבצע ע"י קריאות API רלוונטיות או לחלופין, העתקת קבוצות של נקודות ממקור גרפי כלשהו (קובץ המכיל אותיות למשל). מקור הטקסט יהיה מבנה הנתונים אשר בתוך אובייקט "**אזור הצגת טקסט**".
      - כתגובה לבקשה מן האפליקציה לשנות את ערך הטקסט המוצג: שנוי ערך הטקסט בתוך אובייקט "**אזור הצגת טקסט**" ולאחר מכן ציור הטקסט הרלוונטי ל אובייקט "**אזור הצגת טקסט**" בתוך תחום הנקודות הרלוונטי אליו, ע"ג חלון האפליקציה. ציור הטקסט יתבצע ע"י קריאות API רלוונטיות או לחלופין, העתקת קבוצות של נקודות ממקור גרפי כלשהו (קובץ המכיל אותיות למשל). מקור הטקסט יהיה מבנה הנתונים אשר בתוך אובייקט "**אזור הצגת טקסט**".

דוגמא:

מפת גרפית (DSkin) עבור **אזור הצגת טקסט**:

[ראה איור 15]

אזור הצגת טקסט כפי שנראה באפליקציה עובדת:

[ראה איור 16]

סכמה כללית למימוש "כפתורים לחיצים" (Push Buttons) גמישים בעזרת רעיון Dynamic Skin

1. יצירת / שימוש בשלושה קבצי תמונה שווים בגודלם:
   - קובץ מצב "לחוץ" (נבחר בשם קובץ של **PSkin** : PSkin = Pressed Skin) – הקובץ מתאר (בציור \ תמונה) את חלון האפליקציה כאשר כל הכפתורים בו לחוצים.
   - קובץ מצב רגיל/"לא לחוץ" (נבחר בשם קובץ של **Skin**) הקובץ מתאר (בציור \ תמונה) את חלון האפליקציה כאשר כל הכפתורים בו אינם לחוצים. זהו למעשה חלון האפליקציה במצב הבסיסי שלו.
   - קובץ **המפה הגרפית** (DSkin) המגדיר את האלמנטים הגרפיים, במקרה הזה "כפתורים".
     כפתור, בסכמה זו, מוגדר כאוסף נקודות בעלות צבע מסוים.
     לכל כפתור מוגדר צבע ספציפי, והנקודות הרלוונטיות לכפתור זה (הנקודות הנמצאות באותו מקום של ציור הכפתור בקובץ ה Skin (UnPressedSkin) ובקובץ ה PSkin (PressedSkin) צבועות בצבע זה.
     [הערה: כאן מתוארת התאמה מדוייקת בין הנקודות בקבצי ה: Skin, ב PSkin וב DSkin אולם תיתכן גם התמרה, בין איזורי הצבע בתמונת DSkin **שונה בגודלה**, לבין התמונות האחרות].
     רצוי שקובץ ה DSkin יהיה מאוחסן פורמט גרפי שהנו loseless (אינו גורם לשנוי המידע בתוכו) וזאת בכדי לשמור על החד-ערכיות של איזורי הצבע.
2. המרת הקבצים לפורמט Bitmap וטעינת שלושת הקבצים לזיכרון המחשב.
3. סריקת קובץ ה DSkin וחיפוש איזורי צבע ספציפיים בתוכו:
   3.1. האפליקציה תחפש איזורים בעלי קודים מסוימים של צבע הרלוונטיים לאפליקציה.
   3.2. לאחר הסריקה תוצמד כל קבוצת נקודות השייכות לאותו **אלמנט גרפי** לאובייקט "כפתור" באפליקציה, אובייקט זה יהיה אחראי על ציור הכפתור בכל פעם שיידרש לכך (עדכוני מסך, לחיצות משתמש וכו').
   3.3. אובייקט "כפתור" ישויך לכל אחד מן הצבעים הרלוונטיים ויממש את הפונקציונליות הנדרשת ממנו באפליקציה (למשל ניגון \ עצירת ניגון \ וכו').
   3.4. עם יצירתו יאתחל כל אובייקט "כפתור" ב"אוסף הנקודות" הרלוונטי אליו.
   3.5. כמו כן תחזיק האפליקציה מבנה נתונים שימפה בין קוד הצבע של כל **אלמנט גרפי** לבין אובייקט ה"כפתור" הרלוונטי לצבע זה.
4. הפעלת האפליקציה:
   4.1. לאחר טעינת האפליקציה מתבצעות אחת או יותר מן הפעולות הבאות (וזאת בהתאם לפעולות המשתמש ולמימוש הספציפי של האפליקציה), וכן כל פעולה אחרת המערבת את הנקודות הרלוונטיות ושאותה יכול להגדיר כותב האפליקציה:
   [הערה: **כפתור** יוגדר כאן בתור קבוצת נקודות **ע"ג המסך** המתאימות לקבוצת נקודות ב DSkin – ההתאמה במקרה הנוכחי היא לפי המיקום של הנקודות, כלומר נקודות ב PSkin וב Skin ייחשבו לכפתור במידה והן נמצאות באזור הצבוע בצבעו של הכפתור ב DSkin]
   - כתגובה לבקשת עדכון מסך: העתקת קבוצת הנקודות המתאימה לכל כפתור (כפי שהוגדרה והושמה בסריקה) מאחת מהתמונות PSkin או Skin אל חלון האפליקציה, וזאת לפי מצב הכפתור. מצב הכפתור מאוחסן באובייקט ה"כפתור" הספציפי.
   - כתגובה ללחיצת המשתמש על כפתור שאינו לחוץ: העתקת קבוצת הנקודות המתאימה לכל כפתור (כפי שהוגדרה והושמה בסריקה) מתמונת PSkin אל המסך.
   - כתגובה ללחיצת המשתמש על כפתור לחוץ ("שחרור" הכפתור): העתקת קבוצת הנקודות המתאימה לכל כפתור (כפי שהוגדרה והושמה בסריקה) מתמונת Skin אל המסך.

לדוגמא :

Skin Image
[ראה איור 17]

PSkin Image
[ראה איור 18]

DSkin Image
[ראה איור 19]

תביעות

1. שיטה לחולל ממשק למשתמש באפליקציית מחשב, המכיל את השלבים הבאים:
הגדרת אוסף של אלמנטים גרפיים המוצגים על מסך המחשב;
סריקת מאפייני האלמנטים הגרפיים ע"מ ליצור מפה גרפית של המסך;
הגדרת פונקציות של הממשק למשתמש באפליקציה;
ומיפוי המאפיינים במפה הגרפית לפונקציות של הממשק למשתמש, כך שכל פונקציה מופעלת ע"י בחירה של המשתמש של האלמנט הגרפי בעל המאפיינים הממופים לאותה פונקציה.

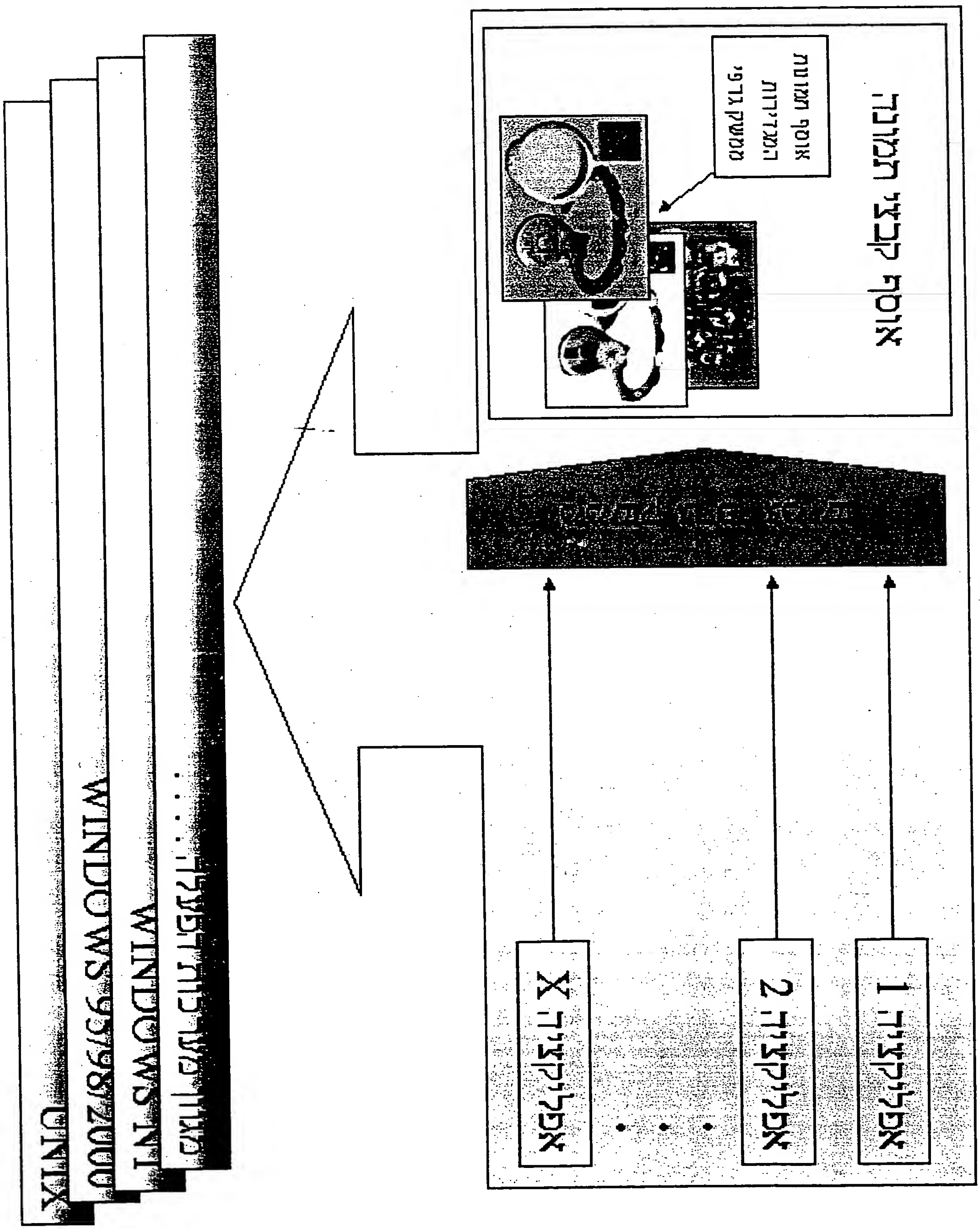
2. שיטה עפ"י תביעה מס' 1, כך שמיפוי המאפיינים לפונקציות נשמר גם כאשר עיצוב ומיקום האלמנטים הגרפיים משתנה.

3. שיטה עפ"י תביעה מס' 1 או 2, כך שתהליך המיפוי אינו תלוי במערכת ההפעלה של המחשב או באפליקציה.

For the Applicant,

Sanford T. Colb & Co.
C:35810

איור מס' 1


אוסף קבצי תמונה
אוסף תמונות
המגדירות
ממשק גרפי
אפליקציה 1
אפליקציה 2
אפליקציה X
מגוון מערכות הפעלה.....
WINDOWS NT
WINDOWS 95/98/2000
UNIX

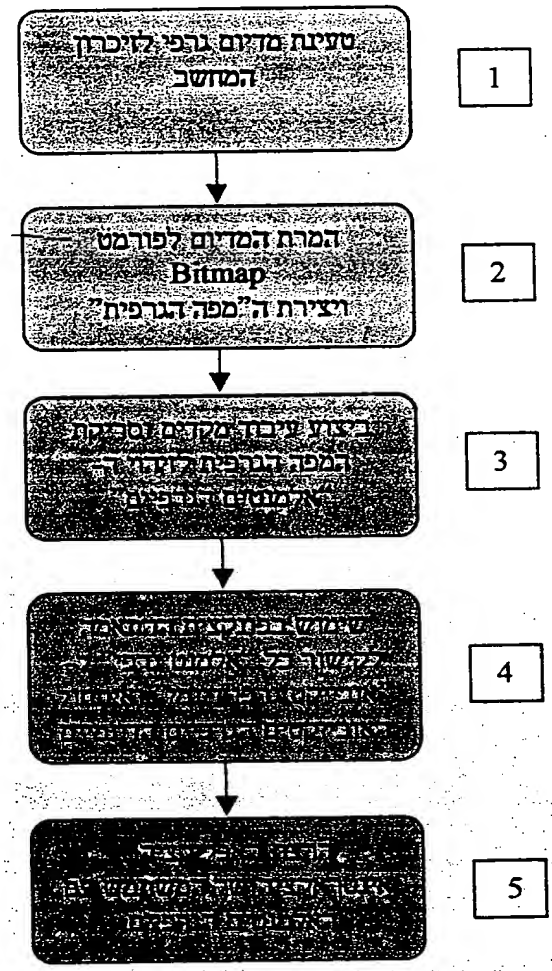
איור מס' 2


מפה גרפית
אפליקציה
קבוצת אלמנטים גרפיים .
למשל:
1 -מתומן זהוב
2 -אזור ירוק
3 - כוכב בעל5 זרועות
4 -משולש שווה שוקיים
פונקצית ההתאמה
קבוצת אובייקטים פונקציונליים
בתוך האפליקציה .למשל:
Play- 1
Stop- 2
Forward- 3
Backward- 4

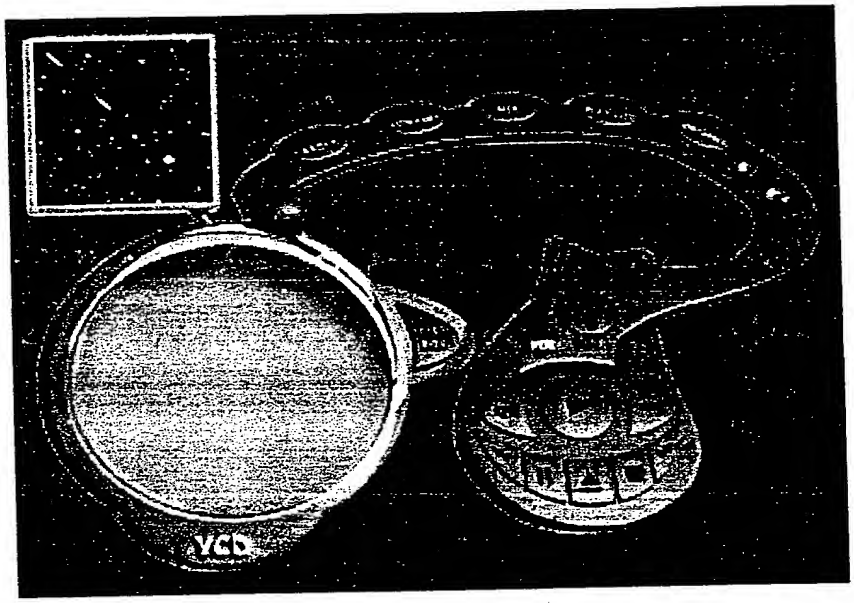
איור מס׳ 3

סכמה כללית של אפליקציה בה משולב Dynamic Skin

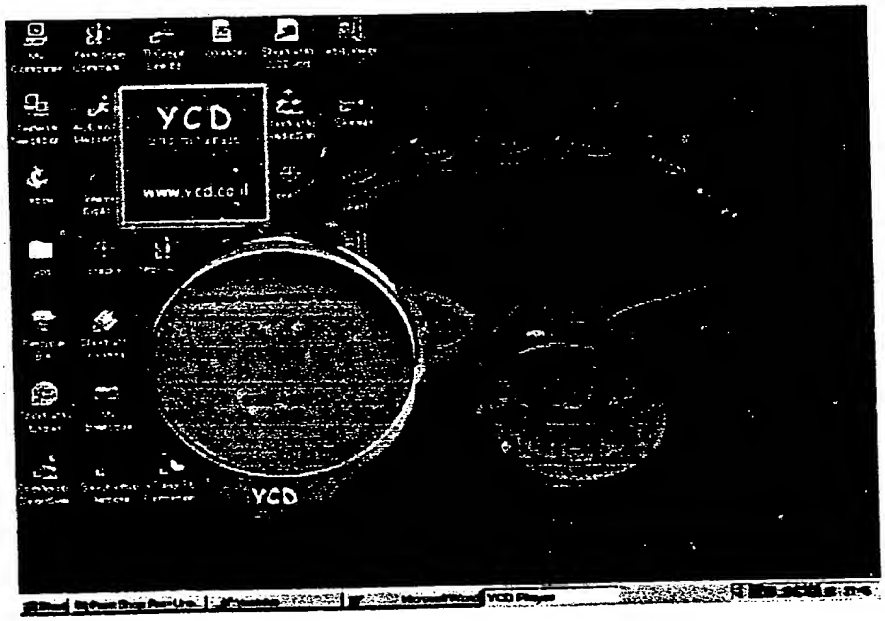
(דיאגרמת מלבנים)



איור מס' 4

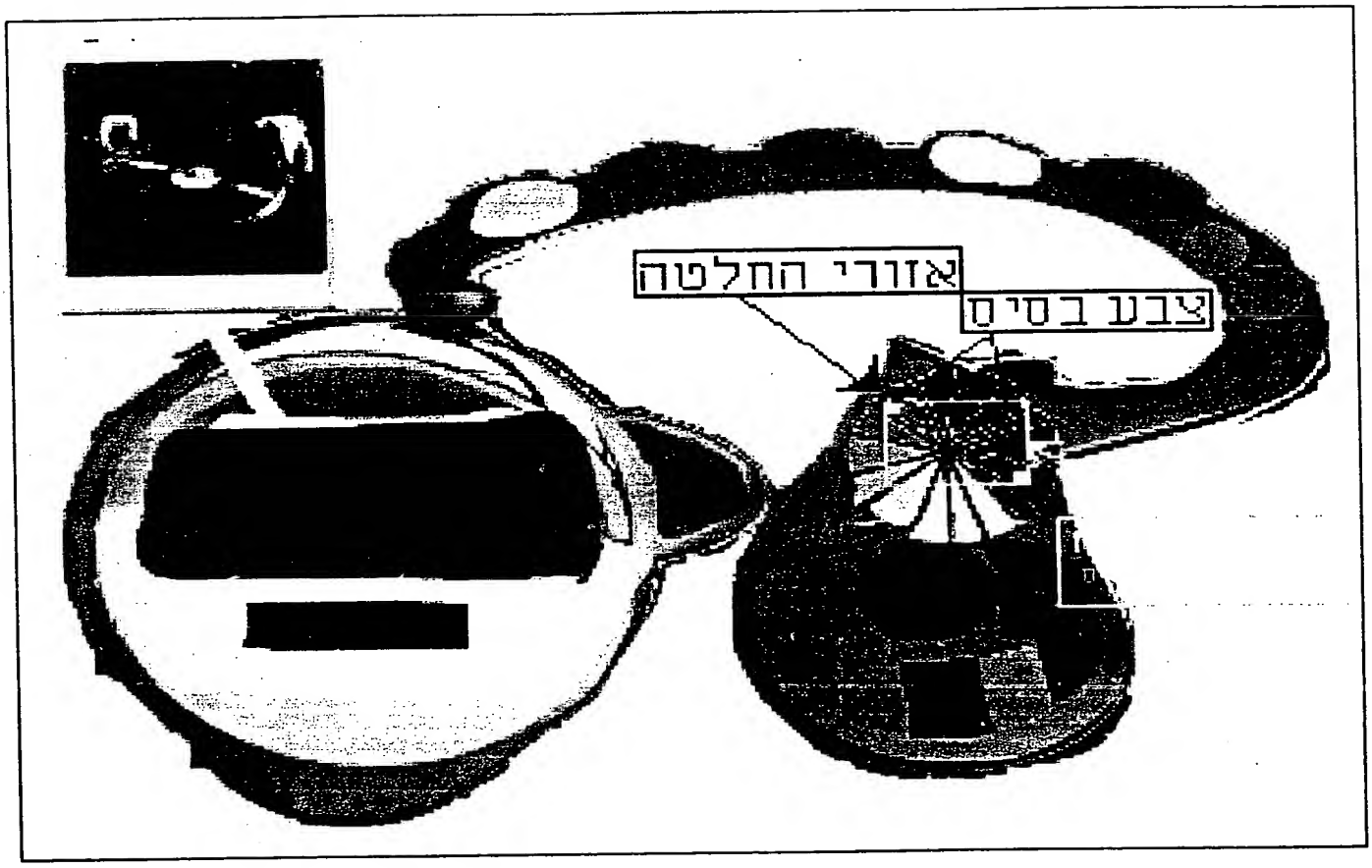


איור מס' 5

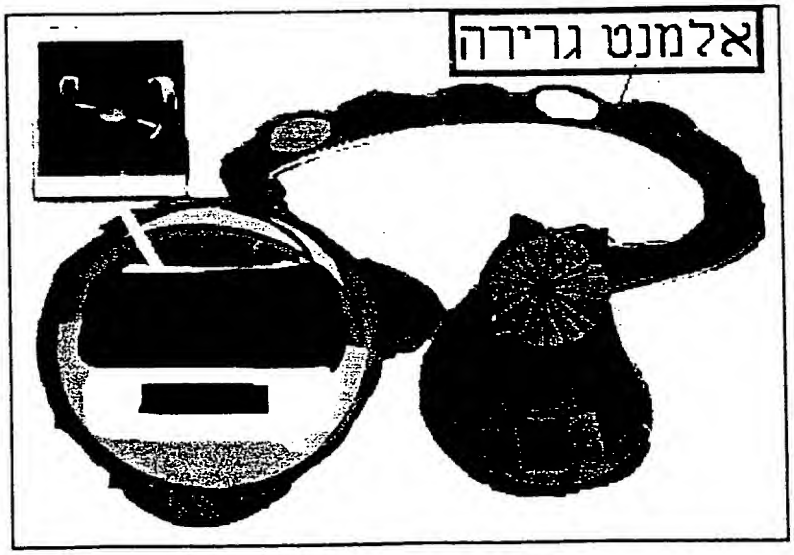


איור מס' 6

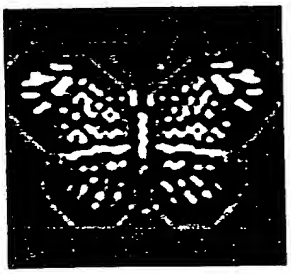
איור מס' 7



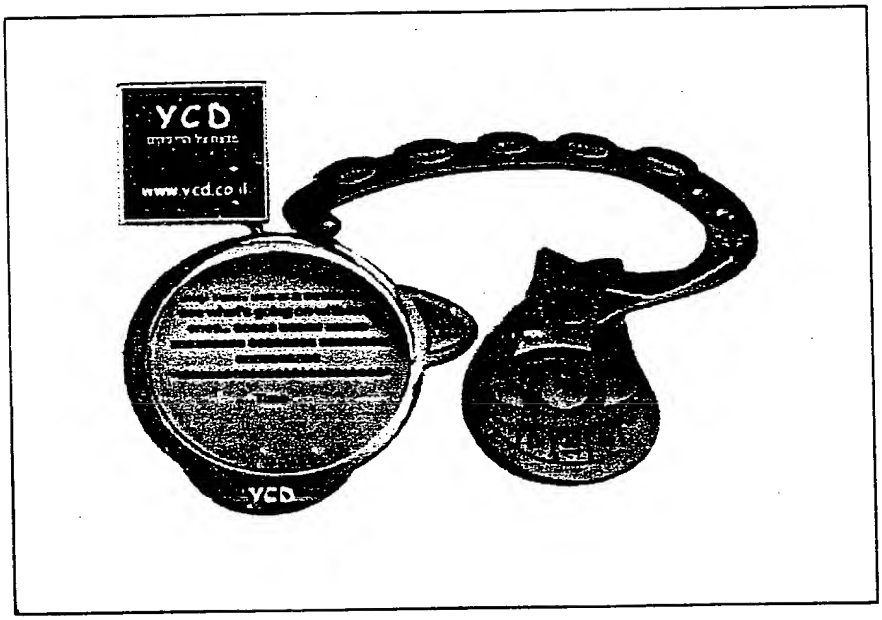
איור מס' 8



איור מס' 9



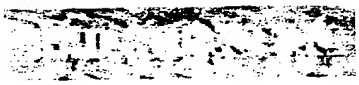
איור מס' 10



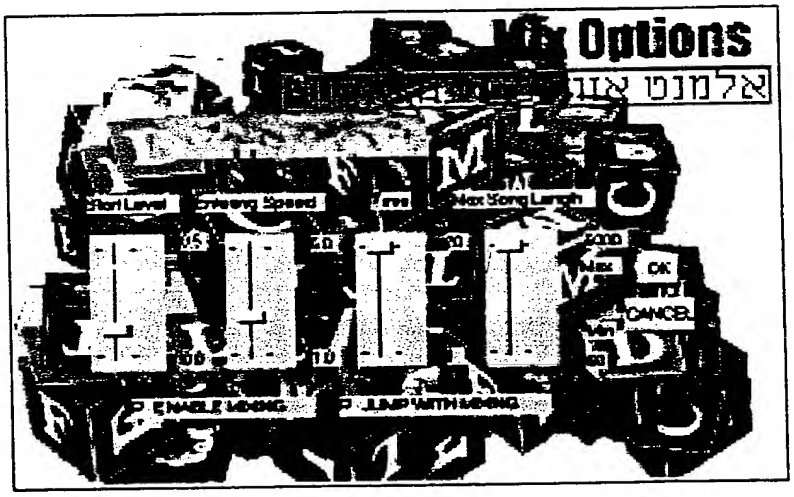
איור מס' 11

איור מס' 12

איור מס' 13

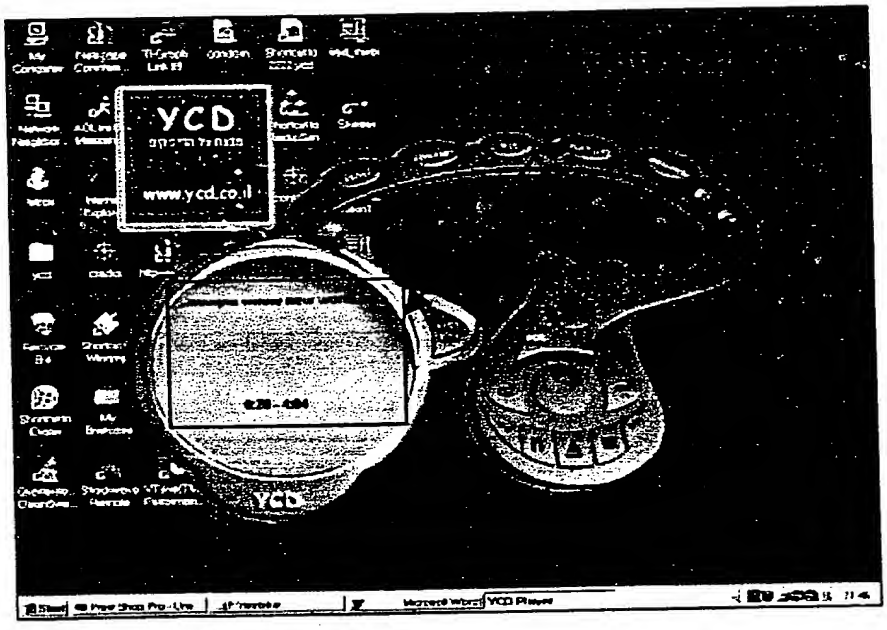


איור מס' 14

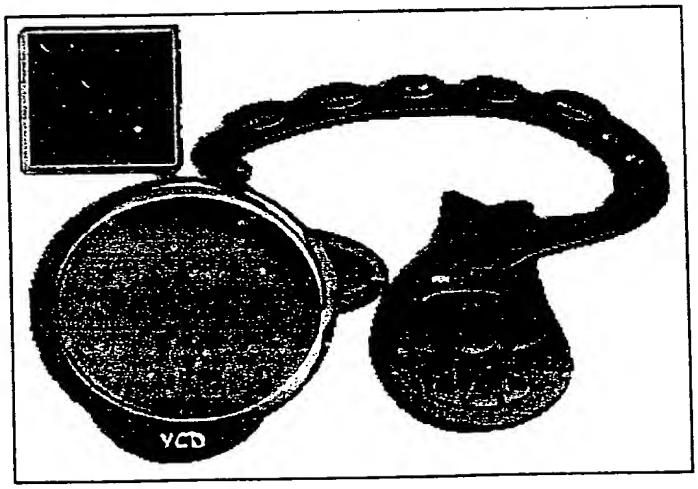


איור מס' 15

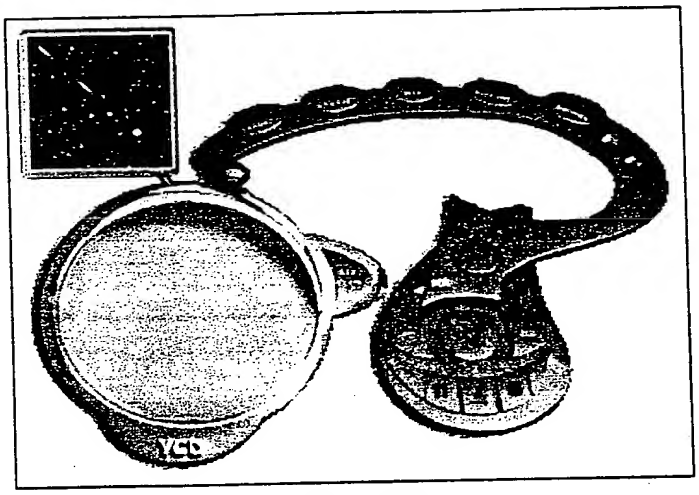
איור מס' 16



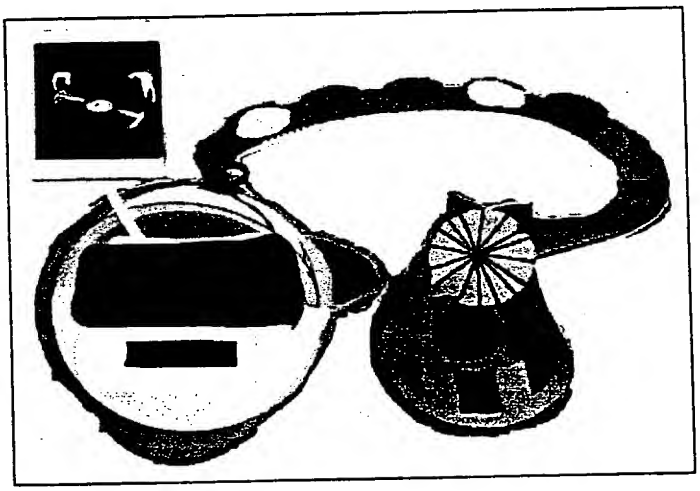
איור מס' 17



איור מס' 18



איור מס' 19



איור מס' 20

$$GE \subseteq GM$$

או:

$$GE \subseteq P(GM)$$

או:

$$GE \subseteq P(P(GM))$$

.

.

$$GE \subseteq P(P(...(P(GM))))$$

איור מס' 21

$$GE \in GM$$